Decision Trees

Fundamentals, Algorithms, and Applications

Ahmed Badi

Mathematics & Machine Learning Enthusiast

ahmedbadi905@gmail.com

December 2025

Outline

Introduction to Decision Trees

Why Decision Trees?

Definition

Decision trees are interpretable, non-parametric models that recursively partition data to make predictions through a series of yes/no questions

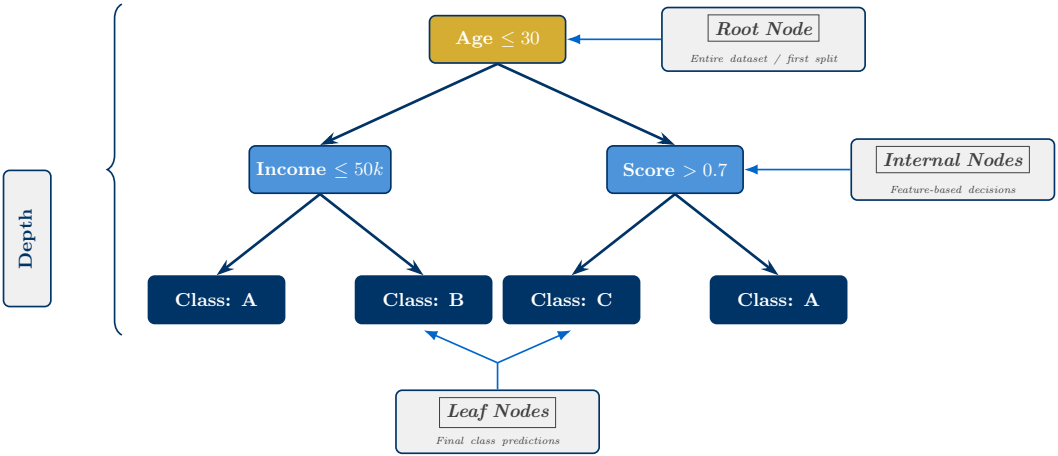
Real-World Examples

- Medical diagnosis
- Credit approval
- Email spam filtering
- Customer segmentation

Key Advantages

- Interpretable results
- Handles both classification and regression
- No feature scaling needed
- Captures non-linear relationships

Tree Anatomy



Tree Components:

- **Root**: starting node
- **Branches**: conditions
- **Internal nodes**: splits
- **Leaf nodes**: predictions

Properties:

- **Depth**: max distance to leaf
- **Balance**: left/right symmetry
- **Size**: number of nodes

How Decision Trees Work

Classification vs Regression Trees

Property	Classification	Regression
Target	Discrete classes	Continuous values
Leaf output	Class label (majority)	Mean/median value
Impurity measure	Gini, Entropy	Variance
Loss function	Misclassification	MSE

Recursive Partitioning Algorithm

Greedy Top-Down Approach

1. Start at root with all data
2. For each node: find best split on one feature
3. Partition data into left/right subsets
4. Repeat recursively on child nodes
5. Stop when stopping criteria met

Stopping Criteria

- Max depth reached
- Min samples per leaf satisfied
- Impurity below threshold
- No improvement from split

Prediction Process

Example: Should I go to the beach?

Decision path:

1. Is weather sunny? $\rightarrow$ Yes
2. Is temperature $> 25C$? $\rightarrow$ Yes
3. Is humidity $< 80\%$? $\rightarrow$ Yes
4. **Prediction: Go!**

Why intuitive?

- Each split is interpretable
- Feature importance visible
- No hidden parameters
- Easy to explain to non-technical users

Splitting Criteria

Entropy and Information Gain

Entropy: Measure of Disorder

$$H(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

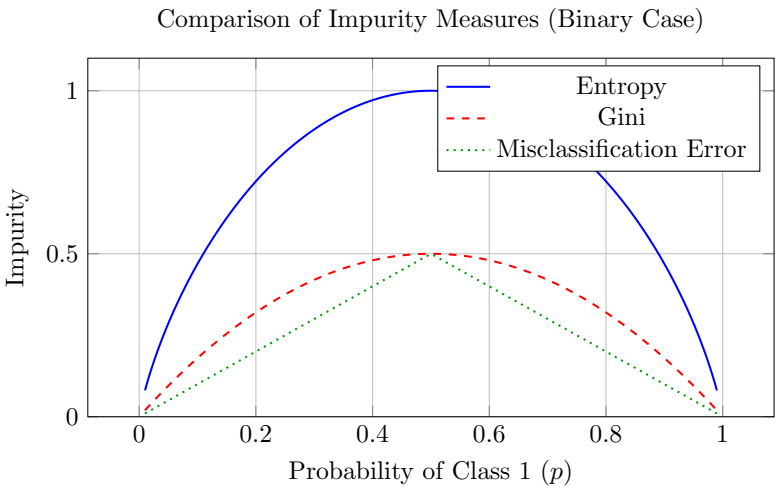
where p_i is fraction of class i in set S

Information Gain

$$IG(S, A) = H(S) - \sum_{v} \frac{|S_v|}{|S|} H(S_v)$$

Choose attribute A that maximizes information gain (entropy reduction)

Impurity


Comparison of Impurity Measures (Binary Case)
Entropy
Gini
Misclassification Error
Impurity
1
0.5
0
0
0.2
0.4
0.6
0.8
1
Probability of Class 1 (p)

Gini Impurity

Gini Index

$$G(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Probability of incorrect classification if random label assigned

Entropy vs Gini

- Both measure purity; results usually similar
- Gini: computationally faster
- Entropy: information-theoretic interpretation
- scikit-learn default: Gini

Regression: Variance Reduction

Variance Reduction

$$VR(S, A) = \text{Var}(S) - \sum_{v} \frac{|S_v|}{|S|} \text{Var}(S_v)$$

where $\text{Var}(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} (y_i - \bar{y})^2$

For Regression Trees

Split to minimize weighted variance of resulting children. Each leaf predicts mean value of its samples.

Algorithms and Pruning

Tree Algorithms Comparison

Algorithm	Criterion	Split Type	Pruning
ID3	Information gain	Binary	N/A
C4.5	Gain ratio	Multi-way	Cost-complexity
CART	Gini impurity	Binary	Cost-complexity

Modern practice: Most implementations use CART or variants (scikit-learn, XGBoost)

The Overfitting Problem

Deep trees overfit

- Perfect training accuracy but poor test performance
- Memorizes noise in training data
- High variance, low bias
- Complex decision boundaries

Solution: Control Complexity

Limit tree depth or leaf samples $\rightarrow$ bias-variance tradeoff

Pre-Pruning (Early Stopping)

Stop Growing Before Overfitting

Common hyperparameters:

- **max_depth**: Maximum tree depth
- **min_samples_split**: Minimum samples to split node
- **min_samples_leaf**: Minimum samples in leaf
- **min_impurity_decrease**: Stop if impurity decreases below threshold

Advantage

Fast and computationally efficient

Post-Pruning: Cost-Complexity

Grow Full Tree, Then Prune

Minimize cost function:

$$R_\alpha(T) = R(T) + \alpha|T|$$

where $R(T)$ is error on validation set, $|T|$ is tree size, α is complexity penalty

Process

1. Generate sequence of trees via α values
2. Evaluate each on validation set
3. Select tree with best validation performance

Features and Importance

Feature Types

Numerical Features

- Threshold-based splits
- Binary: $X \leq t$ vs $X > t$
- Algorithm searches optimal threshold
- Example: Age ≤ 30

Categorical Features

- One-hot encoding or grouping
- Binary splits or multi-way
- Example: Color $\in$ {red, blue}
- Can require many splits

Feature Importance

Impurity-Based Importance

$$\text{Importance}_j = \frac{1}{|T|} \sum_{\text{nodes}} 1(\text{split on } j) \times \Delta\text{Impurity}$$

Measures total impurity decrease from splits on feature j

Advantages and Limitations

Advantages

Interpretability

- White-box model
- Easy explanation
- Feature importance clear
- Suitable for regulated domains

Practical

- No feature scaling
- Fast prediction
- Handles non-linearity
- Mixed data types OK

Limitations

Model Issues

- Prone to overfitting
- Unstable (small data changes $\rightarrow$ big tree changes)
- Biased toward high-cardinality features
- Axis-aligned splits only

Data Issues

- Sensitive to class imbalance
- Struggles with linear patterns
- High variance
- Needs sufficient data per leaf

Conclusion

Key Takeaways

1. Trees recursively partition data into pure regions
2. Gini and entropy measure split quality
3. Overfitting controlled via pruning and hyperparameters
4. Highly interpretable for stakeholders
5. Foundation for powerful ensemble methods (Random Forests, Gradient Boosting)
6. Best when interpretability is critical

When to Use Decision Trees

Good Fit

- Need interpretability
- Mixed feature types
- Non-linear patterns
- Regulatory compliance
- Moderate datasets

Consider Ensembles

- High accuracy needed
- Large complex data
- Ensemble methods
- Random Forests
- Gradient Boosting

Thank You!

Questions & Discussion

Ahmed Badi

Decision Trees: Fundamentals, Algorithms, Applications

ahmedbadi905@gmail.com

December 2025

Backup Slides

ID3 Algorithm (Detailed)

Input: Dataset S, attributes A

1. If all examples same class $\rightarrow$ return leaf
2. If A empty $\rightarrow$ return majority class
3. Best attribute $A^* = \arg\max IG(S, A)$
4. Create node for A^*
5. For each value v of A^*:
   - Create branch
   - Recursively call on S_v (subset where $A^* = v$)

Advantages: Simple, theoretically justified
Disadvantages: Greedy, no pruning, only categorical features

CART Algorithm

Classification and Regression Trees

1. Binary splits only (left/right)
2. Uses Gini impurity for classification
3. Uses variance for regression
4. Exhaustive search: all attributes $\times$ all thresholds
5. Choose split minimizing weighted impurity

Advantages

- Binary splits simpler to interpret
- Handles continuous variables naturally
- Works for both classification and regression
- Cost-complexity pruning available

Cross-Validation for Trees

Stratified k-Fold for Classification

1. Divide into k folds maintaining class proportions
2. Train tree on $k-1$ folds, test on 1
3. Repeat k times
4. Average performance metrics

Hyperparameter Tuning

- Grid search: try combinations of max_depth, min_samples_split, etc.
- Evaluate each via cross-validation
- Select hyperparameters with best CV score
- Retrain on full training set

References

- Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*
- Breiman, L., et al. (1984). Classification and Regression Trees. Chapman and Hall
- Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning
- James, G., et al. (2021). Introduction to Statistical Learning
- Scikit-learn Decision Tree Documentation

Contact

Author: Ahmed Badi

Email: ahmedbadi905@gmail.com

LinkedIn: linkedin.com/in/badi-ahmed

Date: December 2025

Open to collaborations